ยืน ภู่วรวรรณ

เทคนิคการเขียนโปรแกรม ด้วยคลิปเปอร์

เขียนโปรแกรมด้วยคลิปเปอร์จากระดับพื้นฐานคอมพายเลอร์สมบูรณ์แบบมาตรฐาน

ในฉบับก่อนได้กล่าวถึงการคอมพายล์และลิงค์ไปแล้ว ในฉบับนี้จะเน้นไปในเรื่องของการเขียนโปรแกรมโดยเริ่มจากระดับพื้นฐานเพื่อให้เห็นว่าคลิปเปอร์ได้พัฒนาก้าวหน้าออกไป มีการใช้หลักการของภาษาคอมพิวเตอร์อย่างสมบูรณ์แบบ ใช้ไวยากรณ์ลักษณะคล้ายภาษาซีหลายเรื่องมาใช้ ผู้เขียนโปรแกรมจะได้เรียนรู้เทคนิคการเขียนโปรแกรมในรูปแบบภาษาซีได้อีกด้วย

การกำหนดรูปแบบประโยคคำสั่ง

แต่เดิมดีเบสกำหนดประโยครูปแบบคำสั่งโดยใช้คำสั่งแต่ละคำสั่งอยู่ในบรรทัดเดียว แต่ต่อมาในดีเบสทรีมีการใช้ ; บอกว่าประโยคนั้นยังมีต่ออีกบรรทัดหรือบรรทัดถัดต่อมาเป็นบรรทัดที่เชื่อมต่อกันดังรูปที่ 1

```
REPLACE sample1 WITH "data for field sample 1" ;
        sample2 WITH "data for field sample 2"
```

รูปที่ 1 การกำหนดรูปแบบคำสั่ง

โดยปกติเราใช้ย่อหน้า (indent) เข้าช่วยเพื่อให้เห็นโครงสร้างการเชื่อมต่อระหว่างบรรทัดได้ชัดเจนขึ้น ทำให้โครงสร้างโปรแกรมดูสวยงามชัดเจนในการตรวจสอบ

คลิปเปอร์ใช้รูปไวยากรณ์ที่ยอมให้นำเอาประโยคคำสั่งหลายประโยคอยู่ในบรรทัดเดียวกัน โดยใช้เครื่องหมาย ; เป็นตัวคั่นกลาง ทำให้เขียนได้กะทัดรัดขึ้น เช่น

```
i=1 ; j = 2 ; k = 3
```

การเขียนโปรแกรมด้วยคลิปเปอร์ใช้รูปแบบโปรแกรมโครงสร้างเหมือนปาสคาลหรือซี มีการย่อหน้าเพื่อให้ดูชัดเจน คอมพายเลอร์จะไม่สนใจช่องว่าง

หรือบรรทัดว่าง การวางรูปของโปรแกรมจะใช้ลักษณะเป็นฟังก์ชันหรือกระบวนความ ซึ่งจะได้กล่าวถึงในรายละเอียดต่อไป

การเขียนคอมเมนต์

แต่เดิมในดีเบสใช้ * เป็นตัวอักษรแรกของบรรทัดเพื่อแสดงว่าบรรทัดนั้นเป็นบรรทัดคอมเมนต์ หลังจากนั้นต่อมาก็มีการใช้ && เพื่อกำหนดส่วนคอมเมนต์ ในคลิปเปอร์เองก็ยังคงให้ใช้ * ตัวอักษรแรกเป็นบรรทัดคอมเมนต์ และใช้ && หรือ // แทนส่วนคอมเมนต์ที่แทรกในบรรทัดดังตัวอย่างนี้

```
* This line is comment line
a = 5   && assign 5 to a
b = 8   // assign 8 to b
```

หากต้องการเขียนคอมเมนต์ในลักษณะเป็นกลุ่มบรรทัดหรือข้อความโดยไม่จำเป็นต้องเขียนด้วย * ขึ้นบรรทัดหรือเขียนด้วย && ซึ่งจะทำไม่ได้ ในกรณีต้องแทรกคอมเมนต์ในบรรทัดจะใช้รูปแบบการคอมเมนต์ในลักษณะ /* เริ่มต้น และปิดด้วย */ ดังตัวอย่างเช่น

```
/*
   Program name : demo1.prg
   Author       : Piti Pipatpon
   Date         : 12/09/91
   Approved by  :
*/
```

การใช้คอมเมนต์ในลักษณะ /*...*/ นี้ เมื่อแทรกอยู่ในบรรทัดคอมพายเลอร์จะตรวจสอบและดึงออกเองดังนี้

```
a = find_max(/* pass two parameters */
    x,y)
```

ในส่วนสำคัญของโปรแกรมที่ต้องการคอมเมนต์ และต้องการชี้แนะให้เห็นเป็นกลุ่มก้อนที่ชัดเจน หรือกำหนดรายละเอียดบางอย่างของฟังก์ชัน

```
* PRG..... : messages
* CLASS.... : message box
* PREFIX... : msg
* EXPORT... : msg( )
* USES..... : splrtes
*           : windows
*             colors
*             video stack
*             shadows
```

รูปที่ 2 ส่วนสำคัญของโปรแกรมที่ต้องการคอมเมนต์

เรามักจะแสดงคอมเมนต์ด้วย * ที่ตัวบรรทัดมากกว่าจะใช้ /* */ เพราะจะทำให้โปรแกรมเด่นชัดและมองเห็นได้ง่าย ดังรูปที่ 2

การคอมเมนต์ด้วย * นี้ทำให้รูปแบบเด่นชัด เมื่อซอร์สโปรแกรมได้รับการพิมพ์ออกมาก็ตรวจสอบได้ง่าย

ค่าคงตัวที่ใช้ในคลิปเปอร์

ค่าคงตัวที่ใช้ในคลิปเปอร์มีได้หลายชนิดและใช้ได้ในโปรแกรมเช่น เป็นสตริงของตัวอักษร ตัวเลข ลอจิคัล หรือเป็นอะเรย์ ดังรูปที่ 3

ค่าคงตัวที่เป็นตัวเลขจะมีค่าเป็นจำนวนเต็มหรือเป็นจุดทศนิยมก็ได้ แต่จะใช้ค่าคงตัวที่เขียนแบบวิทยาศาสตร์ไม่ได้ สำหรับตัวอักษรนั้นคลิปเปอร์ถือว่าเป็นสตริงหมด โดยไม่มีการแยกแยะเหมือนภาษาซีที่เป็นตัวอักษรตัวเดียวได้ ดังนั้นสตริงในความหมายของคลิปเปอร์คือตัวอักษรเป็นกลุ่มตั้งแต่ตัวเดียว หากนำสตริงมาบวกกันจะหมายถึงการนำมาเชื่อมต่อถึงกัน

การกำหนดค่าตัวอักษรตัวเดียวในลักษณะสตริง ก็ทำได้โดยใช้ CHR( ) โดยกำหนดได้แม้จะเป็นค่าตัวอักษรที่ไม่สามารถพิมพ์ออกมาบนจอได้เช่น chr(13) ตัวอักษรนี้จะนำมาใช้ในโปรแกรมได้

สำหรับเรื่องวันที่ค่าคงตัวนั้นไม่มี แต่สามารถกำหนดเป็นสตริงแล้วใช้ฟังก์ชัน ctod เปลี่ยนค่า เช่น ถ้าใช้ ctod(date( )) จะได้ค่าวันนี้หรือกำหนด x = "10/09/91" แล้วเปลี่ยนเป็นวันที่โดยใช้ ctod(x) การทำงานจะได้ผลลัพธ์เป็นวันที่

Literal

คำสั่งของคลิปเปอร์บางคำสั่งต้องการข้อความที่เป็นตัวอักษรที่บอกรายละเอียดในลักษณะอาร์กิวเมนต์ ข้อความนี้เราเรียก literal ซึ่งคลิปเปอร์จะแปลความและนำไปใช้ เราใช้ส่วนของ literal นี้ในรูปของนิพจน์หรือตัวแปรไม่ได้เช่น

```
use test
```

test เป็นชื่อแฟ้มซึ่งในคลิปเปอร์ถือว่าเป็น literal ซึ่งจะแทนด้วยตัวแปรไม่ได้ เพื่อให้การทำงานสะดวกขึ้นไวยากรณ์ของโปรแกรมนี้จึงใช้มาโครเข้ามาช่วย มาโครในคลิปเปอร์ก็เหมือนในดีเบสจึงไปใช้ & เป็นโอเปอเรเตอร์แทน เช่น กำหนดให้ x = "test" เมื่อ

```
"press any key when ready"      เป็นค่าคงตัวแบบสตริง
3 หรือ 9.85                     เป็นค่าคงตัวแบบตัวเลข
.T.                             เป็นค่าคงตัวแบบลอจิคัล
{9, 8.5, 10}                    เป็นค่าคงตัวแบบอะเรย์
```

รูปที่ 3 ค่าคงตัวที่ใช้ในคลิปเปอร์

จะใช้แฟ้ม test ก็จะได้ดังนี้

```
use &x
```

การแปลความหมายของมาโครจะนำค่าในตัวแปรนี้มาแทน ซึ่งก็จะได้ test ซึ่งคลิปเปอร์จะทำงานในขณะคอมพายล์

พรีโปรเซสเซอร์ (preprocessor)

พรีโปรเซสเซอร์เป็นเรื่องใหม่สำหรับคลิปเปอร์ เป็นเรื่องที่ใช้ในภาษาซีอยู่แล้ว ไวยากรณ์และหลักการของพรีโปรเซสเซอร์นี้จึงเหมือนกับของภาษาซีมาก การคอมพายล์จะดำเนินการประมวลผลขึ้นนี้เพื่อจัดการเบื้องต้นเกี่ยวกับการกำหนดค่าหรือข้อมูลบางอย่าง จากนั้นจึงคอมพายล์ตามไวยากรณ์เดิมของคลิปเปอร์

หากจะพิจารณากลไกการทำงานในวันนี้พบว่าคลิปเปอร์จะนำซอร์สมาคอมพายล์ตามพรีโปรเซสเซอร์ให้ได้เอาต์พุตในรูปของแฟ้มเอกสารอันหนึ่งแล้วจึงนำแฟ้มนั้นมาคอมพายล์อีกครั้ง โดยปกติคลิปเปอร์จะลบแฟ้มเอกสารที่สร้างขึ้นนี้ไปจึงทำให้ผู้ใช้ไม่มีโอกาสได้เห็น แต่ถ้าต้องการจะเห็นเอาต์พุตในขั้นของการทำพรีโปรเซสเซอร์นี้จะต้องใช้คอมพายล์ด้วยเงื่อนไข /P คลิปเปอร์จะสร้างแฟ้ม .ppo ไว้แล้วไม่ลบทิ้ง เมื่อเสร็จการคอมพายล์แล้วขอเรียกดูได้

คำสั่งที่ใช้ในพรีโปรเซสเซอร์จึงเป็นคำสั่งที่คอมพายเลอร์ตัวจริงไม่รู้จัก แต่จะได้รับการเปลี่ยนให้อยู่ในรูปเอาต์พุตที่จะนำไปคอมพายล์ต่อ นักโปรแกรมคอมพิวเตอร์ที่ใช้ภาษาซีชอบวิธีการของพรีโปรเซสเซอร์มาก ทั้งนี้เพราะมีส่วนช่วยในการกำหนดค่าตัวแปร กำหนดเงื่อนไขของการคอมพายล์ การสร้างมาโครสำหรับคลิปเปอร์เวอร์ชัน 5 นี้มีขีดความสามารถได้เท่าเทียมกับภาษาซี

ขีดความสามารถของพรีโปรเซสเซอร์ของคลิปเปอร์มีดังนี้

- การกำหนดค่าคงตัว
- การรวมแฟ้มเข้ามาคอมพายล์ (include)
- การใช้มาโครในการคอมพายล์
- การกำหนดเงื่อนไขของการคอมพายล์
- ผู้ใช้กำหนดคำสั่งขึ้นเอง

ในการใช้งานนี้จะให้รายละเอียดเพื่อเป็นตัวอย่างทีละหัวข้อ

การกำหนดค่าคงตัว

ผู้เขียนโปรแกรมมีความต้องการใช้ชื่อหรือค่าคงตัวบางอย่างในโปรแกรมเพื่อทำให้โปรแกรมมีลักษณะที่ดูเข้าใจได้ง่าย เช่น กำหนดรหัส 13 เรียกว่า Enter ลองพิจารณาจากตัวอย่างโปรแกรมที่เป็นแบบดั้งเดิมดังนี้

```
* wait for escape
ikey = inkey(0)
DO WHILE ikey ! = 27
        .
        .
        ikey = inkey ( 0 )
ENDDO
```

จะเห็นว่า ESC คือรหัสเลข 27 ซึ่งโปรแกรมจะดูเข้าใจได้ยาก แต่ถ้ากำหนดให้ ESC_KEY เป็น 27 แล้วใช้ DO WHILE ikeyi = ESC_KEY ก็จะดูได้ง่ายขึ้น

การกำหนดค่าคงตัวของคลิปเปอร์ใช้เหมือนภาษาซีคือ #define เช่น ถ้าต้องการกำหนดค่าใด ๆ ก็ใช้รูปฟอร์มดังนี้

```
#define KEY_ENTER 13
#define KEY_ESC   27
#define KEY_PGUP  18
```

หลังจากนั้นในโปรแกรมจะใช้ตัวแปรเหล่านี้แทนค่าที่กำหนดได้เลยเช่นดังรูปที่ 4

```
ikey = inkey(0)
DO CASE
      CASE ikey = KEY_ENTER
            .
            .
      CASE ikey = KEY_ESC
            .
            .
      CASE ikey = KEY_PGUP
            .
            .
      ENDCASE
```

รูปที่ 4 โปรแกรมการใช้ตัวแปรแทนค่าที่กำหนดไว้

การใช้ #include

เช่นเดียวกับภาษาซี คลิปเปอร์มีคำสั่ง #include เพื่อให้ไปอ่านข้อมูลหรือโปรแกรมอื่นที่เขียนเป็นแฟ้มเอกสารไว้ภายนอกเอามาใส่ตำแหน่งที่บอกการรวมเอาไว้ การคอมพายล์จะเกิดขึ้นตามที่ได้เขียนไว้ในแฟ้มที่จะมาใช้ร่วมด้วย

การกำหนดแฟ้มสำหรับนำมาใช้ร่วมนี้ คลิปเปอร์กำหนดให้แฟ้มมีส่วนขยายเป็นดังนี้

```
#include "keys.ch"
```

ในส่วนของแฟ้ม .ch อาจจะมีข้อมูลเกี่ยวกับการกำหนดค่าหรือบางส่วนของโปรแกรมเพื่อให้การเขียนโปรแกรมสะดวกขึ้น เช่น ใน keys.ch ดังตัวอย่างนี้

```
#define K_ENTER 13
#define K_ESC   27
#define K_PGUP  18
```

ส่วนของคลิปเปอร์มีแฟ้ม .ch ให้ผู้เขียนโปรแกรมให้ใช้เป็นแฟ้มเฮดเดอร์สำหรับการเรียกมาใช้รวมอยู่แล้วหลายแฟ้ม เช่น box.ch error.ch set.ch std.ch เป็นต้น

โปรแกรมการเขียนคลิปเปอร์

```
/*****************************************************************
PROGRAM NAME : EXAMPLE.PRG
COMPLIEED BY  : CLIPPER EXAMPLE.PRG
LINKED BY     : RTLINK FI EXAMPLE.OBJ LIB CLIPPER.LIB,EXTEND.LIB
WHEN SET ENVIRONMENT VARIABLES OF DOS BY
   SET PATH = C:\CLIPPER5\BIN;(old path)
   SET LIB = C:\CLIPPER5\LIB
   SET INCLUDE = C:\CLIPPER5\INCLUDE
*****************************************************************
*/

// Includes
#INCLUDE "BOX.CH"
#INCLUDE "MEMOEDIT.CH"
#INCLUDE "INKEY.CH"

// Macro definition
#DEFINE TITLE "THE EXAMPLE OF CLIPPER PROGRAM"
#DEFINE VIEW_TEXT 1
#DEFINE EDIT_TEXT 2
#DEFINE VIEW_DBASE 3
#DEFINE BROWSE_DBASE 4
#DEFINE EXIT_TO_DOS 5

* Declare variables
LOCAL Choice := 1
PUBLIC FileName

SET WRAP ON
SET MESSAGE TO 24

DO WHILE Choice > 0
   CLEAR
   SET COLOR TO N/W
   @ 00,40-LEN(TITLE)/2 SAY TITLE
   SET COLOR TO
   @ 01,00 TO 01,79 DOUBLE
   @ 23,00 TO 23,79 DOUBLE
   @ 06,31,16,52 BOX B_DOUBLE_SINGLE
   @ 07,33 PROMPT "1.View Text File" MESSAGE "View Text File"
   @ 09,33 PROMPT "2.Edit Text File" MESSAGE "Edit Text File"
   @ 11,33 PROMPT "3.View dBase File" MESSAGE "View dBase File"
   @ 13,33 PROMPT "4.Edit dBase File" MESSAGE "Edit dBase File"
   @ 15,33 PROMPT "5.Exit " MESSAGE "Exit to DOS"
   MENU TO Choice

   DO CASE
   CASE Choice = VIEW_TEXT .AND. GetFile()
      DO ViewFile
   CASE Choice = EDIT_TEXT .AND. GetFile()
      DO EditFile
   CASE Choice = VIEW_DBASE .AND. GetFile()
      DO ViewDb
   CASE Choice = BROWSE_DBASE .AND. GetFile()
      DO EditDb
   CASE Choice = EXIT_TO_DOS
      Choice = 0
   ENDCASE
ENDDO
RETURN

FUNCTION GetFile // Get text file
FileName := SPACE(32)
@ 19,20 SAY "ENTER FILE NAME :" GET FileName
READ
FileName = TRIM(FileName)
IF FILE(FileName)
   RETURN (.T.)
ELSE
   @ 20,20 SAY "FILE NOT FOUNDED"
   @ 21,20 SAY "Press any key to continue..."
   ?? CHR(7)
   INKEY(0)
   RETURN (.F.)
ENDIF

PROCEDURE ViewFile // View text file
LOCAL cString := MEMOREAD(FileName)
CLEAR
@ 0,1 SAY "VIEW " + FileName
@ 1,0,24,79 BOX B_DOUBLE_SINGLE
SET CURSOR OFF
cString := MEMOEDIT(cString,2,1,23,78,.F.)
SET CURSOR ON
RETURN

PROCEDURE EditFile// Edit text file
LOCAL cString := MEMOREAD(FileName)
CLEAR
@ 0,1 SAY "EDIT " + FileName
@ 1,0,24,79 BOX B_DOUBLE_SINGLE
cString := MEMOEDIT(cString,2,1,23,78,.T.)
IF !MEMOWRIT(FileName,cString)
   ? "Write error"
ENDIF
RETURN

PROCEDURE ViewDb // View dBase file
LOCAL cRecords := ""
CLEAR
USE (FileName)
@ 0,0,24,79 BOX B_DOUBLE_SINGLE
cRecords := "Number of records : " + LTRIM(STR(RECCOUNT()))
@ 0, 79 - 2 SAY "]"
@ 0,(79 - 2) - LEN(cRecords) - 3 SAY "[" + SPACE(LEN(cRecords) + 2)
@ 0,(79 - 2) - LEN(cRecords) - 1 SAY cRecords
@ 0,2 SAY "[" + TRIM(FileName) + "]"
@ 24,2 SAY "[Use " + CHR(27) + CHR(18) + CHR(26) + ;
" To move through data. (Esc to Exit)]"
DBEDIT(1,1,23,78)
USE
RETURN

PROCEDURE EditDb // Browse dBase file
CLEAR
USE (FileName)
@ 0,2 SAY TRIM(FileName)
@ 24,2 SAY "Use " + CHR(27) + CHR(18) + CHR(26) + ; // display arrow keys
" To move through data. Enter : Edit Esc : Exit"
BROWSE(1,1,23,78)
USE
RETURN
```

มาโครสำหรับคอมพายเลอร์

มาโครก็เป็นอีกส่วนหนึ่งที่คลิปเปอร์ใช้โครงสร้างภาษาซี เพื่อกำหนดฟังก์ชันพื้นฐานง่าย ๆ ขึ้นใช้ในส่วนของพรีโปรเซสเซอร์ การใช้ส่วนมาโครนี้ทำให้เกิดความสะดวกและคล่องตัวในการเขียนโปรแกรมได้มากดังตัวอย่างนี้

```
#define MIN(a,b) icf(a<b, a,b)
```

เพียงเท่านี้ก็หมายถึงการกำหนดฟังก์ชัน MIN(a,b) ขึ้นไว้ใช้งานได้แล้ว ผู้เขียนโปรแกรมจะใช้คำสั่งในโปรแกรมดังนี้

```
x = MIN(y,z)
```

รายละเอียดของการใช้มาโครมีเทคนิคได้มากมายจะขอกล่าวในรายละเอียดในโอกาสต่อไป

เงื่อนไขของการคอมพายล์

บางครั้งผู้เขียนโปรแกรมอาจกำหนดให้คอมพายเลอร์ทำงานตามเงื่อนไขบางอย่าง โดยทำการตรวจสอบสถานะบางอย่าง ถ้ามีการกำหนดค่าไว้ใน #define ให้คอมพายล์อย่างหนึ่งถ้าไม่มีการกำหนดค่าไว้ให้คอมพายล์โปรแกรมอีกส่วนหนึ่ง คำสั่งที่ใช้ในการกำหนดเงื่อนไขของการคอมพายล์มีหลายคำสั่งดังรูปที่ 5

เมื่อใช้งานเราจะใช้โครงสร้างการเขียนโปรแกรมดังตัวอย่างนี้

```
#define TEST_VERSION
หลังจากนั้นจะมีการทดสอบ
#ifdef TEST_VERSION
#endif
```

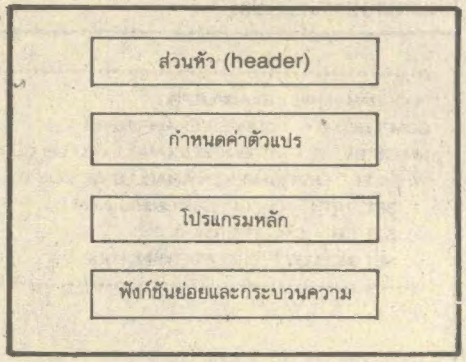
หากจะให้มีเงื่อนไข #else ก็แทรกได้ดังนี้

```
#define TEST
#ifdef TEST
     } โปรแกรม
#else
     } โปรแกรม
#endif
```

คลิปเปอร์จะเลือกคอมพายล์ ถ้ามีประโยค #define TEST ก็จะคอมพายล์ส่วนบน ถ้าตัดประโยค #define TEST ออกก็จะคอมพายล์ส่วนล่าง

โครงสร้างคลิปเปอร์แบบฟังก์ชันและกระบวนความ

โปรแกรมคลิปเปอร์เป็นโปรแกรมที่ใช้รูปโครงของภาษาซีอยู่มาก ในเวอร์ชัน 5 จึงได้กำหนดโครงสร้างของโปรแกรมหลักเป็นโปรแกรมแรกเรียกคอมพายล์ โดยไม่ต้องขึ้นต้นด้วย PROCEDURE หรือฟังก์ชัน แต่ส่วนโปรแกรมย่อยอื่นจะใช้โครงสร้างเป็นแบบ PROCEDURE และ FUNCTION



รูปที่ 6 โครงสร้างของโปรแกรมที่เขียนด้วยคลิปเปอร์

โครงสร้างของโปรแกรมจะมีลักษณะดังรูปที่ 6

เพื่อพิจารณาตัวอย่างโปรแกรมจริงและการทดลองที่จะเกิดขึ้นจากผู้เริ่มต้นลองใช้โปรแกรมนี้เป็นตัวอย่างพื้นฐาน โดยขอให้ดูรายละเอียดวิธีการกำหนดส่วน header การกำหนดค่าตัวแปรในฟังก์ชัน กระบวนความและโปรแกรมหลัก เพื่อดูว่าการกำหนดนั้นมีการกำหนดตัวแปรเป็นแบบท้องถิ่นและ global ได้อย่างไร

ส่งท้าย

การเขียนโปรแกรมด้วยคลิปเปอร์เวอร์ชัน 5.0 นี้เป็นแบบอย่างที่ดีสำหรับผู้ศึกษาเพราะรูปแบบได้ยึดหลักมาตรฐานสากลทั่วไป โดยเฉพาะโครงสร้างภาษาซีทั้งหมดจึงนับเป็นโอกาสอันดีสำหรับผู้สนใจใฝ่หาความรู้จะได้นำการทดลองที่เสนอนี้มาประยุกต์ใช้งานในอนาคตต่อไป

#ifdef	ทดสอบว่ามีการกำหนดค่าใน define หรือไม่
#ifndef	ทดสอบว่าถ้าไม่มีการกำหนดใน define
#else	ใช้รวมกับ #if #ifdef หรือ #ifndef
#endif	เป็นคำสั่งที่จบส่วนการทำงาน

รูปที่ 5 คำสั่งที่ใช้ในการกำหนดเงื่อนไขของคอมพายล์